INFORMATION AND COMPUTATION 110, 391–420 (1994)

A Foundational Delineation of Poly-time

DANIEL LEIVANT*

Department of Computer Science,
Indiana University, Bloomington, Indiana 47405

We show that a function over $\{0, 1\}^*$ is poly-time iff it is computed by an equational program which can be proved to be everywhere converging in constructive second-order logic with set-existence (comprehension) restricted to positive quantifier-free formulas, or alternatively with set-existence for positive existential formulas. These set-existence principles convey an ontology of infinite sets as evolving, not completed, totalities. Our characterization results can consequently be viewed as a foundational justification for identifying poly-time with feasibility.

1. INTRODUCTION

Machine-independent characterizations of computational complexity classes, such as poly-time, lend further credence to the importance of the classes considered, provide insight into their nature, relate them to issues relevant to programming methodology and to program verification, suggest new tools for separating complexity classes, and offer concepts and methods for generalizing computational complexity to computing over arbitrary structures and to higher type functionals. Machine independent characterizations fall, by and large, into three major groups: database queries (i.e., global methods of finite model theory), applicative programs over free algebras, and proof-theoretic characterizations.

Feasible computing has long been identified with computability within deterministic polynomial time, primarily on practical and circumstantial grounds. Most known worst-case lower-bounds are either polynomials of small degrees, which are clearly feasible, or are at least exponential, which are clearly non-feasible. Moreover, poly-time functions are easily defined and computed and are closed under many natural operations. However, the central importance of poly-time has been contested lately notably because feasible probabilistic classes might subsume poly-time in their

* I am grateful to Steve Bellantoni, Sam Buss, Steve Cook, Jean-Yves Girard, Alex Ignjatovic, Bruce Kapron, Yves Lafont, and Jean-Yves Marion for helpful comments about earlier versions of this work. E-mail address for the author: leivant@cs.indiana.edu.

0890-5401/94 $6.00

practical significance and because super-polynomial bounds such as $n^{\log\log n}$ are lower than, say, n^{100}, for all n encountered in practice ($n < 2^{2^{100}}$). It is, therefore, of particular interest that characterizations of poly-time abound within each of the three groups of machine-independent characterizations. Within the descriptive approach, poly-time queries are characterized by recursion equations [Saz80, Gur83], by positive first-order fixpoints [Var82, Imm86], by first-order inflationary fixpoints [GS86, Lei90a], and by alternating transitive-closure [Imm87]. Applicative characterizations of poly-time exist in terms of subrecursive schemas [Cob65, BC92, Lei93] and of representability in certain typed λ-calculi [LM93].

Of special interest are proof theoretic characterizations, because proof principles codify directly conceptual abstraction and are therefore the most natural medium for exploring and articulating a foundational justification for linking feasibility with poly-time. Proof theoretic characterizations of classes of computable functions follow the following pattern. Given a formalism F whose language provides a natural rendition of statements of the form "*program P converges for all input*," one associates with F the class $C(F)$ of computable functions for which there exists a program that F proves to converge for all input. Such characterizations have traditionally used number theories. For instance, Schütte and Kreisel showed that the provably recursive functions of first-order arithmetic (classical or constructive) are precisely the ε_0-recursive functions (see, e.g., [Kre60a, Kre65]), and Parsons showed that the functions provably recursive in arithmetic with existential induction are precisely the primitive recursive functions [Par77]. Continuing this line of research, Buss showed that the poly-time functions are precisely the provable functions of bounded arithmetic, a weak number theory which uses a particularly restricted form of induction [Bus86].

Unfortunately, characterizations that use weak number theories still lack some of the foundational traits we are after, because within a number theory one cannot examine directly arbitrary computable functions, in a way that will sort out the feasible from the non-feasible ones. On the one hand, some functions are presumed feasible from the outset, without foundational justification, such as addition, multiplication, or Cobham's function $\#$ ($x \# y =_{\mathrm{df}} 2^{(\log x)(\log y)}$). On the other hand, direct reference to non-feasible functions is excluded from the outset, because any function admitted is trivially provable.

Thus, we seek a proof theoretic characterization of poly-time that presupposes no functions and makes no assumption about properties of particular functions. That is, we wish to refer to a formalism in which one can reflect on the concept of number itself, by allowing variables to range over entities other than numbers. We use here pure second-order logic as such a formalism. More precisely, we use second-order logic to

reason about computations over $\{0, 1\}^*$, rather than over $\mathbb{N}$, because computational complexity is traditionally calibrated for computing over $\{0, 1\}^*$.

The use of second-order logic is motivated by the fact that sets like $\mathbb{N}$ and $\{0, 1\}^*$ are second-order definable, allowing one to reason about computations over such sets, including divergent computations, without recourse to coding or to non-logical axioms. By using a standard formalism for second-order logic, we thus obtain a framework for delineating classes of computable functions: the natural spectrum of set-existence (comprehension) principles used in the second-order formalism gives rise to a spectrum of corresponding classes of computable functions, thereby linking conceptual abstraction (as reflected by set-existence) to computational complexity. Moreover, certain set-existence principles can be seen to convey particular levels of ontological commitments. Therefore, second-order logic is a vehicle for formalizing the correspondence between philosophically anchored viewpoints (as reflected by levels of set-existence) and computational complexity classes. In particular, we identify here a correspondence between poly-time and levels of set-existence that convey the view that infinite sets exist only as potential, not completed, totalities.

We comment further on foundational aspects of the main result after stating it precisely, in Section 4 below.

2. Coherent Equational Programs

2.1. *Equational Programs over Free Algebras*

The computation model we use here is equational programs, in the Herbrand–Gödel style (see e.g. [Kle52]). This rudimentary model is especially amenable to proof theoretic treatment, since its syntax is already part of the syntax of first-order logic with equality. We use such programs over arbitrary free (term) algebras, in particular the free algebra $\mathbb{N}$ generated from a constant $\mathbf{0}$ and a unary function $\mathbf{s}$, and the free algebra $\mathbb{W}$ generated from a constant $\boldsymbol{\varepsilon}$ and unary functions $\mathbf{0}$ and $\mathbf{1}$. Note that $\mathbb{W}$ can be identified with the set $\{0, 1\}^*$: for example, we identify the word 011 with the term $\mathbf{011}\boldsymbol{\varepsilon} = \mathbf{0}(\mathbf{1}(\mathbf{1}(\boldsymbol{\varepsilon})))$, which we call the *term for* 001 (we treat other words similarly).

Programs are defined as follows. Given a free algebra $\mathbb{A}$, we posit an unlimited supply of r-ary *program-functions* ($r = 1, 2, \ldots$), which are identifiers distinct from the constructors of $\mathbb{A}$. The *terms* are generated from the constructors of $\mathbb{A}$, free variables, and program functions. A *statement* over $\mathbb{A}$ is an equation between terms. A set of statements is a *program-body*. A *program* over $\mathbb{A}$ is a pair $(P, \mathbf{f})$, where P is a program-body and $\mathbf{f}$ is a program-function, called the program's *principal identifier*. We write V_P for

the vocabulary of P, i.e., the set consisting of the constructors of $\mathbb{A}$ and the program-functions used in P.

Given a program-body P, write $P \vdash E$ if E is a V_P-equation derivable from P in equational logic. That is,

1. $P \vdash E$ for every $E \in P$;
2. $P \vdash \mathbf{t} = \mathbf{t}$ for every V_P-term $\mathbf{t}$;
3. if $P \vdash E$ then $P \vdash [\mathbf{t}/u]E$ for every V_P-term $\mathbf{t}$ and variable u (where $[\mathbf{t}/u]$ is the operation of substituting $\mathbf{t}$ for all occurrences of u);
4. if $P \vdash [\mathbf{t}/u](\mathbf{s} = \mathbf{q})$ and $P \vdash \mathbf{t} = \mathbf{t}'$, then $P \vdash [\mathbf{t}'/u](\mathbf{s} = \mathbf{q})$.

Note that taking u for $\mathbf{q}$ in (4) yields transitivity, and taking $u = \mathbf{t}$ for $\mathbf{s} = \mathbf{q}$ yields symmetry. We say that P is *coherent* if for every distinct $\mathbf{a}, \mathbf{a}' \in \mathbb{A}$, $P \nvdash \mathbf{a} = \mathbf{a}'$.

Each (r-ary) program-function $\mathbf{g}$ in P induces on $\mathbb{A}$ the relation

$$\mathbf{g}^P \underset{\text{df}}{=} \{(\mathbf{a}_1 \cdots \mathbf{a}_r, \mathbf{b}) \in \mathbb{A}^{r+1} \mid P \vdash \mathbf{g}(\mathbf{a}_1 \cdots \mathbf{a}_r) = \mathbf{b}\}.$$

Obviously, if P is coherent, then each $\mathbf{g}^P$ is univalent, i.e. (the graph of) a partial function. The function *computed* by $(P, \mathbf{f})$ is then the partial function $\mathbf{f}^P$.

2.2. *Some Programs over* $\mathbb{W}$

The following program over $\mathbb{W}$, with principal identifier $\oplus$, computes the concatenation function. We let $\mathbf{c}$ range over $\{\mathbf{0}, \mathbf{1}\}$.

$$\varepsilon \oplus w = w \qquad (\mathbf{c}v) \oplus w = \mathbf{c}(v \oplus w).$$

The program for $\oplus$, augmented with the following equations, computes the function which, on input v, w returns $w^n = w \cdots w$ with $n = \text{length}(v)$ concatenated copies of w.

$$w \odot \varepsilon = \varepsilon \qquad w \odot (\mathbf{c}v) = w \oplus (w \odot v).$$

The destructor (predecessor) function for $\mathbb{W}$ is computed by the program

$$\mathbf{p}(\varepsilon) = \varepsilon \qquad \mathbf{p}(\mathbf{c}u) = u.$$

2.3. *Models for Programs*

Given a program-body P over a free algebra $\mathbb{A}$, we write $\tilde{\forall} P$ for the conjunction of the universal closures of all statements in P. Our canonical model for $\tilde{\forall} P$ is the quotient algebra

$$\mathscr{S}(\mathbb{A}, P) \underset{\text{df}}{=} (\mathbb{A}_P)/(\approx_P),$$

where $\mathbb{A}_P$ is the free algebra generated from V_P, and $\mathbf{t} \approx_P \mathbf{t}'$ iff $P \vdash \mathbf{t} = \mathbf{t}'$. Since $=_P$ is an equivalence relation (indeed a congruence relation), we write $[\mathbf{t}]_\approx$ (or $[\mathbf{t}]$) for the equivalence class of $\mathbf{t}$. For example, if P is the program-body over $\mathbb{N}$ consisting of the single equation $\mathbf{f}(x) = \mathbf{f}(\mathbf{s}(x))$, then the universe of $\mathscr{S}(\mathbb{A}, P)$ is $\{[\mathbf{s}^i\mathbf{f}^j\mathbf{0}] \mid i, j \geqslant 0\}$, where (using regular-expression notation) $[\mathbf{s}^i\mathbf{f}^j\mathbf{0}] = \mathbf{s}^i(\mathbf{fs}^*)^j\mathbf{0}$. We have here infinitely many equivalence-classes corresponding to "divergence," each consisting of infinitely many terms. Note that if $\mathbf{g}^P$ is total for each $\mathbf{g} \in V_P$, then the universe of $\mathscr{S}(\mathbb{A}, P)$ is $\{[\mathbf{a}]_\approx \mid \mathbf{a} \in \mathbb{A}\}$.

If σ is a substitution of closed V_P-terms for variables, then we write $[\sigma]$ for the environment over $\mathscr{S}(\mathbb{A}, P)$ defined by $[\sigma]x = [\sigma x]$. By induction on terms, we have $[\sigma]\mathbf{t} = [\sigma\mathbf{t}]$, for every V_P-term $\mathbf{t}$.

LEMMA 2.1. *For every algebra $\mathbb{A}$ and program P over $\mathbb{A}$ we have $\mathscr{S}(\mathbb{A}, P) \models \tilde{\forall} P$.*

Proof.

$\mathbf{t} = \mathbf{t}' \in P \Rightarrow P \vdash \sigma\mathbf{t} = \sigma\mathbf{t}'$ — all substitutions σ of $\mathbb{A}_P$-terms, by definition of $\vdash$

$\Rightarrow [\sigma\mathbf{t}]_\approx = [\sigma\mathbf{t}']_\approx$ — all such σ, by definition of $\approx$

$\Rightarrow [\sigma]\mathbf{t} = [\sigma]\mathbf{t}'$ — by definition of $[\sigma]$

$\Rightarrow \mathscr{S}(\mathbb{A}, P), [\sigma] \models \mathbf{t} = \mathbf{t}'$ — by definition of $\models$

$\Rightarrow \mathscr{S}(\mathbb{A}, P), \eta \models \mathbf{t} = \mathbf{t}'$ — all environments η in $\mathscr{S}(\mathbb{A}, P)$ since for every η there is a substitution σ such that η and $[\sigma]$ agree on $\mathbf{t}$ and $\mathbf{t}'$

$\Rightarrow \mathscr{S}(\mathbb{A}, P) \models \mathbf{t} = \mathbf{t}'$. ∎

There are alternative natural modelings for a coherent program-body P. One is the *partial* model of $\tilde{\forall} P$, consisting of $\mathbb{A}$ with each $\mathbf{g} \in V_P$ interpreted as the partial function $\mathbf{g}^P$. However, partial structures, in which function-identifiers are interpreted as partial functions, are not in the mainstream of model theory or proof theory. Another construction is the flat domain $\mathbb{A}_\perp$ whose universe is $\mathbb{A}$ augmented with an object $\perp$ (intended to denote divergence), and with each constructor $\mathbf{c}$ of $\mathbb{A}$ interpreted "strictly," i.e., as the function that sends $a_1 \cdots a_r$ (where $r = arity(\mathbf{c})$) to $\mathbf{c}(a_1 \cdots a_r)$, if $a_1 \cdots a_r \in \mathbb{A}$, and to $\perp$, if some a_i is $\perp$. It would seem plausible to consider the expansion $\mathbb{A}_\perp^P$ of $\mathbb{A}_\perp$, in which a q ary $\mathbf{g} \in V_P$ is interpreted as

$$g(x_1 \cdots x_q) = \begin{cases} y & \text{if } \mathbf{g}^P(x_1 \cdots x_q) = y \\ \perp & \text{otherwise (including if some } x_i \text{ is } \perp). \end{cases}$$

However, $\mathbb{A}_\perp^P$ need not be a model of $\tilde{\forall}P$. Consider the program $(P, \mathbf{f})$ over the one-element algebra $\mathbb{U} = \{\varepsilon\}$, with $P = \{\mathbf{f}(x, y) = x\}$. Then $\mathbb{U}_\perp^P \models \mathbf{f}(\varepsilon, \perp) = \perp$ by strictness. If $\mathbb{U}_\perp^P \models \tilde{\forall}P$, then $\mathbb{U}_\perp^P \models \mathbf{f}(\varepsilon, \perp) = \varepsilon$, and so $\mathbb{U}_\perp^P \models \varepsilon = \perp$, contradicting the definition of $\mathbb{U}_\perp$.

2.4. *Completeness of Coherent Programs*

It is well-known that equational programs over the algebra $\mathbb{N}$ are complete for μ-recursive functions.[1] It is easy to infer from this that for every free algebra $\mathbb{A}$ and every computable function f over $\mathbb{A}$ there is an equational program that computes f. However, it is useful here to simulate directly the operation of a deterministic TM by an equational program over $\mathbb{W}$.

We refer to a variant of TM's, operating over a finite tape consisting of *cells* separated by *borders*, and whose size varies during the computation.[2] The empty tape consists of a single border. It is useful to think of the scanning head as being always positioned at a border. The transitions are of the following forms.

Branching. $sq_0q_1q_\varepsilon$, with the operational meaning being that in state s, if the character to the right (of the head) is 0 or 1, then switch to state q_0 or q_1, respectively; if there is no character to the right (i.e., end of tape), then switch to state q_ε.

Insertion. $s\mathbf{c}q$: in state s, insert $\mathbf{c}$ to the right of the head, and switch to state q.

Deletion. sDq: in state s, delete the character to the right of the head (if any), and switch to state q.

Right. sRq: in state s, move the head to the right, if possible, and switch to state q.

Left. sLq: in state s, move the head to the left, if possible, and switch to state q.

Clearly, any standard TM M operating on a single infinite tape can be simulated by a TM of the type above, operating on the non-blank portion

[1] See, e.g., [Kle52]. A slightly simplified construction is this. Given $f(\vec{x}) = \mu y.\, g(\vec{x}, y) = \mathbf{0}$, where g is primitive recursive, f is defined by supplementing the primitive-recursive definition of g by $\mathbf{a}(u, \mathbf{s}v) = u$; $\mathbf{b}(\vec{x}, \mathbf{0}) = \mathbf{g}(\vec{x}, \mathbf{0})$, $\mathbf{b}(\vec{x}, \mathbf{s}u) = \mathbf{a}(\mathbf{g}(\vec{x}, \mathbf{s}u), \mathbf{b}(\vec{x}, u))$; $\mathbf{c}(u, \mathbf{0}) = u$; $\mathbf{f}(\vec{x}) = \mathbf{c}(u, \mathbf{b}(\vec{x}, u))$.

[2] These are, in fact, the same as the first-order machines of [Lei87] over the structure $\mathbb{W}$, with two registers (pointers), i.e., a 2-stack machine over $\mathbb{W}$. One referee has noted that such machines bear similarities to Kolmogorov machines and to Schönhage's storage modification machines [Sch80, Gur88].

of M's tape. Moreover, the simulation of each step of M takes a small fixed number of steps (depending on the exact formulation of the TM model to which M belongs). Thus, if M runs in time $O(T(n))$ then so does the simulating machine.

LEMMA 2.2. *Let M be a deterministic TM as above, and let r be defined by*

$$r(w, t) = \textit{the term for the portion of the tape from the head of } M \textit{ and on to the right, after } |t| \textit{ steps of } M\textit{'s run on input } w.$$

Then r is computable by a coherent equational program.

Proof. Let $s_0, \ldots, s_n$ be the states of M, and set a coding for states; say $\#s_i =_{\mathrm{df}} \mathbf{0}^{n-i}\mathbf{1}^i\boldsymbol{\varepsilon}$, $i = 0, \ldots, n$. We define an equational program P_M that computes r, simultaneously with functions l and st, where $l(w, t)$ is the term for the portion of the tape from M's head and on to the left, in reverse order, and $st(w, t)$ is the code of the state of M, both after $|t|$ steps of M's run on input w. The definition of P_M below is not the simplest possible, but it serves an additional purpose later.

The equations in P_M are the defining equations for the destructor function $\mathbf{p}$, plus the following (where $\mathbf{c} = \mathbf{0}, \mathbf{1}$). We use $\mathbf{r}$, $\boldsymbol{l}$, and $\mathbf{st}$ as program-functions for r, l, and st.

- $$\mathbf{r}(w, \boldsymbol{\varepsilon}) = w$$
  $$\mathbf{l}(w, \boldsymbol{\varepsilon}) = \boldsymbol{\varepsilon}$$
  $$\mathbf{st}(w, \boldsymbol{\varepsilon}) = \#s \quad \text{where } s \text{ is the initial state of } M$$

- $$\mathbf{r}(w, \mathbf{c}t) = \mathbf{r}_1(\mathbf{st}(w, t), \boldsymbol{l}(w, t), \mathbf{r}(w, t))$$
  $$\mathbf{l}(w, \mathbf{c}t) = \boldsymbol{l}_1(\mathbf{st}(w, t), \mathbf{l}(w, t), \mathbf{r}(w, t))$$
  $$\mathbf{st}(w, \mathbf{c}t) = \mathbf{st}_1(\mathbf{st}(w, t), \boldsymbol{l}(w, t), \mathbf{r}(w, t))$$

- for each transition $sq_0q_1q_\varepsilon$ of M the equations

$$\mathbf{r}_1(\#s, u, v) = v$$
$$\boldsymbol{l}_1(\#s, u, v) = u$$
$$\mathbf{st}_1(\#s, u, \boldsymbol{\varepsilon}) = \#q_\varepsilon$$
$$\mathbf{st}_1(\#s, u, \mathbf{0}v) = \#q_0$$
$$\mathbf{st}_1(\#s, u, \mathbf{1}v) = \#q_1$$

- for each transition $s\mathbf{c}q$ the equations

$$\mathbf{r}_1(\#s, u, v) = \mathbf{c}v$$
$$\boldsymbol{l}_1(\#s, u, v) = u$$
$$\mathbf{st}_1(\#s, u, v) = \#q$$

- for each transition sDq the equations

$$\mathbf{r}_1(\#s, u, v) = \mathbf{p}v$$
$$\boldsymbol{l}_1(\#s, u, v) = u$$
$$\mathbf{st}_1(\#s, u, v) = \#q$$

- for each transition sRq the equations

$$\mathbf{r}_1(\#s, u, v) = \mathbf{p}v$$
$$\boldsymbol{l}_1(\#s, u, \boldsymbol{\varepsilon}) = u$$
$$\boldsymbol{l}_1(\#s, u, \mathbf{c}v) = \mathbf{c}u$$
$$\mathbf{st}_1(\#s, u, v) = \#q$$

- for each transition sLq the equations

$$\mathbf{r}_1(\#s, \boldsymbol{\varepsilon}, v) = v$$
$$\mathbf{r}_1(\#s, \mathbf{c}u, v) = \mathbf{c}v$$
$$\boldsymbol{l}_1(\#s, u, v) = \mathbf{p}u$$
$$\mathbf{st}_1(\#s, u, v) = \#q$$

- if s is a state for which M has no transition,

$$\mathbf{r}_1(\#s, u, v) = u$$
$$\mathbf{l}_1(\#s, u, v) = v$$
$$\mathbf{st}_1(\#s, u, v) = \#s.$$

Write usv for the machine-configuration with state s, tape uv, and head positioned at the border between u and v. Clearly, if M is a deterministic machine, then the three equations $P \vdash \mathbf{st}_1(\#s, u, v) = \#s'$, $P \vdash \boldsymbol{l}_1(\#s, u, v) = u'$, and $P \vdash \mathbf{r}_1(\#s, u, v) = v'$ hold iff either M has a one-step transition from configuration usv to $u's'v'$, or else no such transition exists and $u's'v' = usv$. By induction on m it follows that M, starting with input w, reaches a configuration usv in m steps iff $P \vdash \boldsymbol{l}(w, \mathbf{0}^m\boldsymbol{\varepsilon}) = u$, $P \vdash \mathbf{r}(w, \mathbf{0}^m\boldsymbol{\varepsilon}) = v$, and $P \vdash \mathbf{st}(w, \mathbf{0}^m\boldsymbol{\varepsilon}) = \#s$. Thus $(P_M, \mathbf{r})$ computes r.

To see that P_M is coherent, consider the structure $\mathscr{W}$ for the vocabulary V_P, obtained by expanding the free algebra $\mathbb{W}$ with the intended interpretations of $\mathbf{r}$, $\mathbf{l}$, $\mathbf{st}$, $\mathbf{r}_1$, $\mathbf{l}_1$, and $\mathbf{st}_1$. These interpretations are functions; they are not over-defined because M is deterministic, and they are not under-defined because the last entry in the list of equations above guarantees that $\mathbf{r}_1$, $\mathbf{l}_1$, $\mathbf{st}_1$, $\mathbf{r}$, $\mathbf{l}$, and $\mathbf{st}$ are defined for all input. Clearly, $\mathscr{W}$ is a model of $\bar{\forall}P$. Therefore, every equation derivable from P is true in $\mathscr{W}$, in particular $\mathbf{w} = \mathbf{w}'$ cannot be derived for distinct $\mathbf{w}$, $\mathbf{w}'$. ∎

3. Second-Order Logic and Computational Complexity

3.1. *Second-Order Delineation of Free Algebras*

We consider second-order logic, obtained by augmenting the syntax of first-order formulas with variables ranging over relations, and quantification over such variables. It is well-known that the natural numbers are second-order definable. That is, the following formula defines, in any structure for a vocabulary containing the constants $\mathbf{0}$ and $\mathbf{s}$, the denotations of the numerals $\mathbf{0}$, $\mathbf{s}(\mathbf{0})$, $\mathbf{s}(\mathbf{s}(\mathbf{0}))$, ...

$$N[x] \underset{\text{df}}{\equiv} \forall Q(Q(\mathbf{0}) \wedge \forall u(Q(u) \to Q(\mathbf{s}(u))) \to Q(x)).$$

More generally, if $\mathbb{A}$ is a free algebra, then the following formula defines in any structure (for a vocabulary containing the constructors of $\mathbb{A}$) the denotations of the $\mathbb{A}$-terms

$$A[x] \underset{\text{df}}{\equiv} \forall Q \quad Cl_{\mathbb{A}}[Q] \to Q(x),$$

where

$$Cl_{\mathbb{A}}[Q] \underset{\text{df}}{\equiv} \bigwedge_{\mathbf{f} \text{ constructor of } \mathbb{A}} Cl_{\mathbf{f}}[Q]$$

$$Cl_{\mathbf{f}}[Q] \underset{\text{df}}{\equiv} \forall u_1 \cdots u_r (\, Q(u_1) \wedge \cdots \wedge Q(u_r) \to Q(\mathbf{f}(u_1 \cdots u_r)) \,).$$

$$(arity(\mathbf{f}) = r \geqslant 0).$$

That is, $Cl_{\mathbb{A}}[Q]$ is a formula stating that Q is closed under the constructors of $\mathbb{A}$. For instance,

$$Cl_{\mathbb{W}}[Q] \underset{\text{df}}{\equiv} Q(\varepsilon) \wedge \forall u(Q(u) \to Q(\mathbf{0}u)) \wedge \forall u(Q(u) \to Q(\mathbf{1}u)).$$

Henceforth we write Cl for $Cl_{\mathbb{W}}$.

3.2. *Convergence*

To formally state the convergence of an equational program for some or for all input, one needs to refer to potentially non-terminating computations. An approach common in proof theory, and due to Kleene [Kle52, Kle69], is to explicitly describe operational convergence in a formalism sufficiently rich to numerically code (Gödelize) the operational machinery. In logics of programs convergence is expressed using modal operators or potentially non-denoting terms. We continue here the alternative approach of [Lei83, Lei90], which might be dubbed *ecto-algebraic*, since variables may potentially range over objects other than algebra elements, the intended models are all super-structures of $\mathbb{A}$, and programs are considered not as definitions of partial functions over the free algebra $\mathbb{A}$ in hand, but as uniform definitions of total functions over any model of $\tilde{\forall}P$. This is in contrast to formalism such as Peano arithmetic, analysis, and bounded arithmetic, which are *endo-algebraic*. We use here second-order logics as ecto-algebraic formalisms in which membership in a free algebra $\mathbb{A}$ is definable. Alternatively, membership in $\mathbb{A}$ can be axiomatized within a first-order theory whose vocabulary includes a unary relational identifier **A** intended to denote membership in $\mathbb{A}$, as in Peano's original formalism for first-order arithmetic [Pea89]; this style is developed in [Lei94].

Among the advantages of ecto-algebraic formalisms are the absence of entities extraneous to the free algebras proper (such as auxiliary functions, computation constructs, special axioms, or inference rules); the ability to refer freely to computationally divergent terms and to partial computable functions; and a tight correspondence between feasible computation and philosophically anchored ontologies of the infinite.

The key observation underlying the ecto-algebraic approach is the following.

THEOREM 3.1. [Convergence Theorem]. *Let* $(P, \mathbf{f})$ *be a coherent equational program over an algebra* $\mathbb{A}$. *The following are equivalent.*

1. *The function* f *over* $\mathbb{A}$ *computed by* $(P, \mathbf{f})$ *is total.*
2. $\models \tilde{\forall}P \wedge A[\vec{x}] \to A[\mathbf{f}(\vec{x})]$, *i.e. in every model* $\mathscr{S}$ *of* $\tilde{\forall}P$ *the set of denotations of* $\mathbb{A}$*-terms is invariant under the interpretation* $\mathbf{f}^{\mathscr{S}}$ *of* $\mathbf{f}$.
3. $\mathscr{S}(\mathbb{A}, P) \models \tilde{\forall}P \wedge A[\vec{x}] \to A[\mathbf{f}(\vec{x})]$.

Here $arity(\vec{x}) = arity(\mathbf{f})$, $A[x_1 \cdots x_k]$ *abbreviates* $A[x_1] \wedge \cdots \wedge A[x_k]$, *and the relational quantifiers have their standard interpretation.*

Proof. Assume (1), consider a model $\mathscr{S}$ of $\tilde{\forall}P$, and suppose that $A[\vec{a}]$ holds for some tuple $\vec{a}$ of elements of the universe of $\mathscr{S}$. By the definition of the formula A, $A[\vec{a}]$ implies that $\vec{a}$ are denotations in $\mathscr{S}$ of some

$\mathbb{A}$-terms $\vec{\mathbf{a}}$. Since $(P, \mathbf{f})$ computes a total function over $\mathbb{A}$, there is some $\mathbf{b} \in \mathbb{A}$ for which $P \vdash \mathbf{f}(\vec{\mathbf{a}}) = \mathbf{b}$, and therefore $\tilde{\forall} P \models \mathbf{f}(\vec{\mathbf{a}}) = \mathbf{b}$ (by soundness of equational logic). For $\mathbf{b} \in \mathbb{A}$ we have $\mathscr{S} \models A[\mathbf{b}]$ trivially. Since $\mathscr{S} \models \tilde{\forall} P$, we therefore get $\mathscr{S} \models A[\mathbf{f}(\vec{\mathbf{a}})]$. We have thus shown that $\mathscr{S} \models A[\vec{x}] \to A[\mathbf{f}(\vec{x})]$, establishing (2).

The implication (2) $\Rightarrow$ (3) is trivial.

Finally, assume (3) and let f be the partial-function over $\mathbb{A}$ computed by $(P, \mathbf{f})$. By Lemma 2.1 $\mathscr{S}(\mathbb{A}, P) \models \tilde{\forall} P$, and so, by (3), $\mathscr{S}(\mathbb{A}, P) \models A[\vec{x}] \to A[\mathbf{f}(\vec{x})]$. For every $\vec{\mathbf{a}} \in \mathbb{A}$ we have $\mathscr{S}(\mathbb{A}, P) \models A[\vec{\mathbf{a}}]$, and therefore $\mathscr{S}(\mathbb{A}, P) \models A[\mathbf{f}(\vec{\mathbf{a}})]$. By definition of $\mathscr{S}(\mathbb{A}, P)$, this implies that there is a $\mathbf{b} \in \mathbb{A}$ such that $\mathbf{f}(\vec{\mathbf{a}}) \approx_P \mathbf{b}$, that is, $P \vdash \mathbf{f}(\vec{\mathbf{a}}) = \mathbf{b}$, and so $f(\vec{\mathbf{a}}) = \mathbf{b}$, since $(P, \mathbf{f})$ computes f. Thus f is total, proving (1). ∎

3.3. *Characterization of Computational Complexity by Set-Existence Principles*

Theorem 3.1 links the convergence of a program $(P, \mathbf{f})$ over $\mathbb{A}$ to the validity (in all structures, not only in $\mathbb{A}$) of the formula $\mathrm{Tot}[P, \mathbf{f}] \equiv (\tilde{\forall} P \wedge A[\vec{x}] \to A[\mathbf{f}(\vec{x})])$. Restricting both sides of this correspondence, we obtain a relation between convergence of $(P, \mathbf{f})$ *within given computational resources* and truth of $\mathrm{Tot}[P, \mathbf{f}]$ *provably within a given higher order logic.* That correspondence is formalized by the following definition. Let L be a formalism for second (or higher) order logic. A function f over $\mathbb{A}$ is *provable in* L iff it is computed by some equational program $(P, \mathbf{f})$ such that

$$\vdash_L \tilde{\forall} P \wedge A[\vec{x}] \to A[\mathbf{f}(\vec{x})].$$

More specifically, we obtain a correspondence between the computational complexity of functions and the *level of abstraction* needed to prove their convergence. Abstraction can be calibrated by *set-existence principles.* If Φ is a set of formulas, then set-existence (or "comprehension") for Φ is the principle of admitting relations defined by formulas in Φ:

$$\exists R\, \forall x_1 \cdots x_k (R(x_1 \cdots x_k) \leftrightarrow \varphi) \qquad \text{for} \quad \varphi \in \Phi \text{ in which } R \text{ is not free.}$$

A proof calculus for second-order logic with set existence for Φ is defined in Section 1 below. We write $\mathrm{IL}_2(\Phi)$ and $\mathrm{ML}_2(\Phi)$ for intuitionistic and minimal second-order logic, respectively, with set-existence for formulas in Φ.

A simple correspondence between set-existence and a class of computable functions falls out of Prawitz's interpretation of second-order arithmetic in second-order logic [Pra65]; namely, the provable numeric functions of second-order logic with set-existence for *all* second-order formulas are precisely the provably recursive functions of second-order

arithmetic. Prawitz's interpretation uses a relativization of first-order quantifiers to N, allowing induction for all objects in the range of the quantifiers: for if $N[x]$, i.e.,

$$\forall Q(Q(\mathbf{0}) \wedge \forall u(Q(u) \to Q(\mathbf{s}(u))) \to Q(x))$$

then, by set existence for φ, we get

$$\varphi[\mathbf{0}] \wedge \forall u(\varphi[u] \to \varphi[\mathbf{s}(u)]) \to \varphi[x].$$

However, to obtain induction for a *numeric* formula φ, one needs to interpret φ using a relativization of quantifiers to N. For example, an existential formula $\exists n\psi$ of arithmetic is interpreted in second-order logic by $\exists x(N[x] \to \psi')$ (for a suitable variant ψ' of ψ). Since N is second-order, set-existence for first-order formulas does not suffice to derive induction, even for (the interpretations of) Σ^0_1 formulas.

An interesting level of set-existence refers to "computational" (*strict-*Π^1_1 [Bar69]) formulas, i.e., formulas of the form $\forall \vec{R}\, \exists \vec{x} \psi$, where $\vec{R}$ are relational variables, and ψ is quantifier-free, with no other relational variables.[3] Then a numeric function f is primitive recursive iff it is provable in second-order logic with set-existence for computational formulas.

Set existence has been viewed for some time as a yardstick for calibrating the abstraction level of mathematical reasoning. Much work has been done on subsystems of analysis, with important results relating levels of set-existence to fundamental theorems of analysis. In particular, the *Reverse Mathematics* project initiated by H. Friedman and pursued by S. Simpson and others has shown that many of the major theorems of analysis are equivalent to certain set-existence principles (see e.g. [Fri75, Sim86]). One important difference between Reverse Mathematics and the present development is that the former uses higher order quantification on top of a basic arithmetic theory, whereas we work within pure second-order logic, a setting that allows us to reach down to more feasible computational complexity classes.

4. Statement and Discussion of the Main Result

4.1. *A Symmetric Rendition of Free Algebras*

From the definition W of $\mathbb{W}$ one can easily derive basic properties of $\mathbb{W}$, such as its closure under the destructor function, and the fact that every $w \in \mathbb{W}$ is obtained using the constructors, i.e.,

$$W[w] \to w = \varepsilon \vee \exists v(W[v] \wedge (w = \mathbf{0}v \vee w = \mathbf{1}v)).$$

[3] The significance of computational formulas and their relation to computability are discussed in [Lei92].

However, the natural proofs of this and similar properties rely on set-existence for complex formulas. With the kind of restricted set-existence we intend to use it seems necessary to also use a formally weaker rendition of $\mathbb{W}$ which, intuitively, uses as basic operations on $\mathbb{W}$ not only the constructors, but also the destructor and the definition-by-cases functions. The usefulness of a more symmetric definition of $\mathbb{W}$ is not surprising, because computation models are normally symmetric with respect to construction and destruction of the data objects; e.g., the definition of Turing machines treats symmetrically the move of the scanning head to the right or to the left.

For a unary relation variable Q, let

$$Cl^+[Q] \underset{\text{df}}{\equiv} \forall u(Q(u) \to Q(\mathbf{0}u)) \wedge \forall u(Q(u) \to Q(\mathbf{1}u))$$

$$Bk[Q, u] \underset{\text{df}}{\equiv} u = \varepsilon \vee \exists v(Q(v) \wedge u = \mathbf{0}v) \vee \exists v(Q(v) \wedge u = \mathbf{1}v)$$

$$Cl^-[Q] \underset{\text{df}}{\equiv} \forall u\, Q(u) \to Bk[Q, u]$$

$$\widehat{Cl}[Q] \underset{\text{df}}{\equiv} Q(\varepsilon) \wedge Cl^+[Q] \wedge Cl^-[Q]$$

$$\widehat{W}(x) \underset{\text{df}}{\equiv} \forall Q\, \widehat{Cl}[Q] \to Q(x).$$

Thus $Cl^-[Q]$ implies that Q is closed under the destructor (predecessor) function, and $\widehat{W}$ may be viewed as a symmetric definition of the algebra $\mathbb{W}$. An analogous rendition can be defined for any free algebra.

Since $\widehat{Cl}[Q] \to Cl[Q]$ holds trivially, we have $W[x] \to \widehat{W}[x]$. The converse, $\widehat{W}[x] \to W[x]$, can be proved using set-existence for the formula W itself, as follows.

LEMMA 4.1. *$\widehat{W}[x] \to W[x]$ is provable in minimal second-order logic with comprehension for universal second-order formulas.*

Proof. Arguing within minimal second-order logic, assume $\widehat{W}[x]$. Instantiating the main quantifier to the formula $W[u]$ we obtain $\widehat{Cl}[W] \to W[x]$. We prove the premise, concluding $W[x]$. $W[\varepsilon]$ and $W[w] \to W[\mathbf{c}w]$ $(\mathbf{c} = \mathbf{0}, \mathbf{1})$ are straightforward.

To verify $Cl^-[W]$, assume $W[v]$, and show $Bk[W, v]$. Instantiating the main quantifier in $W[v]$ to the relation $Bk[W, u]$, we have

$$Bk[W, \varepsilon] \wedge \forall u(Bk[W, u] \to Bk[W, \mathbf{0}u]) \wedge \forall u(Bk[W, u] \to Bk[W, \mathbf{1}u]) \to Bk[W, v].$$

We have $Bk[W, \varepsilon]$ trivially, and since $W[u] \to W[\mathbf{c}u]$, we easily get $Bk[W, u] \to Bk[W, \mathbf{c}u]$. Thus we conclude $Bk[W, v]$, proving the lemma. ∎

Since W and $\widehat{W}$ are semantically equivalent (with respect to the standard interpretation of relational quantifiers), we have from Theorem 3.1:

THEOREM 4.2. *Let $(P, \mathbf{f})$ be an equational program over $\mathbb{W}$, computing a partial function f over $\mathbb{W}$. Then f is total iff $\models \tilde{\forall} P \wedge W[\vec{x}] \to \widehat{W}[\mathbf{f}(\vec{x})]$.*

We say that a function f is *weakly provable* (*w-provable*) in L, if there is a program $(P, \mathbf{f})$ for f such that

$$\tilde{\forall} P \vdash_L W[\vec{x}] \to \widehat{W}[\mathbf{f}(\vec{x})].$$

Since W trivially implies $\widehat{W}$ (without use of set existence), it follows that provability is formally at least as strong a property as w-provability. Also, note that the composition of a provable function with a w-provable function is w-provable: if $\forall x(W[x] \to W[\mathbf{f}(x)])$ and $\forall y(W[y] \to \widehat{W}[\mathbf{g}(y)])$, then $\forall x(W[x] \to \widehat{W}[\mathbf{g}(\mathbf{f}(x))])$.

4.2. *A Characterization Theorem for Poly-time*

A formula is *positive* if it contains no negation or implication. Let QF^+ and Σ_1^+ denote the collections of positive quantifier-free formulas and positive existential formulas, respectively. Our main characterization theorem is

THEOREM 4.3. *Let L_2 be one of IL_2 or ML_2, and let Φ be one of QF^+ or Σ_1^+. A function f over $\mathbb{W}$ is computable in deterministic polynomial time iff f is w-provable in $L_2(\Phi)$.*

Preliminary forms of this work (including [Lei91]) referred to slightly different variants of w-provability and of $\mathrm{IL}_2(QF^+)$. The muted role of existential quantification in the theorem is not unexpected: if a function is provably recursive in Peano arithmetic with Σ_1^0 induction then it is provably recursive already with Δ_1^0 induction [Par77].

Theorem 4.3 easily follows from the following two lemmas.

LEMMA 4.4. *Every poly-time function over $\mathbb{W}$ is w-provable in $\mathrm{ML}_2(QF^+)$.*

LEMMA 4.5. *If a computable function f over $\mathbb{W}$ is w-provable in $\mathrm{IL}_2(\Sigma_1^+)$, then it is poly-time.*

We prove these lemmas in the next two sections. We shall show elsewhere that, in fact, Lemma 4.5 and Theorem 4.3 remain true also for $\mathrm{CL}_2(\Sigma_1^+)$.

Proof of Theorem 4.3. If f is poly-time then it is w-provable in $\mathrm{ML}_2(QF^+)$, by Lemma 4.4. Since this provability is within the weakest of the logics considered, all other provability properties follow trivially.

Conversely, suppose a function f over $\mathbb{W}$ is w-provable in $L_2(\Phi)$ (for L_2 and Φ as in the theorem's statement). Then it is w-provable in $\mathrm{IL}_2(\Sigma_1^+)$, since $\mathrm{IL}_2(\Sigma_1^+)$ is the strongest one among the logics considered. By Lemma 4.5 this implies that f is poly-time. ▮

While Theorem 4.3 shows that the w-provable functions are closed under composition, this is not at all obvious from the definition of w-provability, which refers to different properties for the input and the output. We shall discuss elsewhere a more symmetric variant of provability, for which Theorem 4.3 remains true, and for which closure under composition is immediate from its definition.

4.3. *A Foundational Interpretation of the Main Theorem*

We submit that set-existence for positive quantifier-free and positive existential formulas is conceptually related to feasibility, and that, consequently, Theorem 4.3 establishes a foundational justification for identifying feasibility with poly-time.

The unfeasibility of exponentiation is evident already when considering short expressions with no physical realization,[4] such as $5^{5^{5^{55}}}$. Evidently, one cannot show in practice that this term has a numeric value by calculating it; instead, one proves that the exponentiation function yields, in general, numerically meaningful results from numerically meaningful arguments. That proof is by induction.

However, the use of induction here has an impredicative aspect, as pointed out by Isles [Isl92]. Recall that a definition of a set $S \subset \mathbb{N}$ is *impredicative* if it refers (e.g., via quantification over sets) to $\mathscr{P}\mathbb{N}$: since S is itself a member of $\mathscr{P}\mathbb{N}$, the definition is circular. Such impredicative definitions are accepted under the common stipulation that the collection $\mathscr{P}\mathbb{N}$ is well-delineated in advance. Predicative mathematics is based on rejecting that assumption and replacing it by a progression of predicatively justifiable sub-collections of $\mathscr{P}\mathbb{N}$ (see, e.g., [Fef69, Kre60]). Nelson [Nel86] has pointed out that the predicative critique can, in fact, be applied already to uses of induction over $\mathbb{N}$, i.e., to the admission of $\mathbb{N}$ as a completed totality. Induction can be viewed as an implicit definition of $\mathbb{N}$: whereas the conditions $\mathbf{0} \in \mathbb{N}$ and $x \in \mathbb{N} \Rightarrow \mathbf{s}(x) \in \mathbb{N}$ provide a lower bound for the extension of $\mathbb{N}$, induction provides an upper bound. But the proof by induction that exponentiation is well-defined over $\mathbb{N}$ presupposes that addition is well-defined for *arbitrary* elements of $\mathbb{N}$, i.e.,. that $\mathbb{N}$ is already obtained as a completed totality. Thus, the implicit definition of $\mathbb{N}$

[4] The number given here greatly exceeds the volume of the universe measured in the volume of an electron, multiplied by the age of the universe measured in the traversal time of light through a distance equal to the radius of an electron.

by induction is circular. It follows that a genuinely predicative reasoning about $\mathbb{N}$ (or other free algebras) should regard infinite sets as *evolving* entities (a well known theme in constructive mathematics, dating back to Brouwer), rather than as completed totalities. This acceptance of objects as they are constructed, and not on the basis of a speculative acceptance of an entire infinite totality, seems to indeed convey a predicative/feasible mathematical ontology.

What set existence principle corresponds, then, to this predicative approach? That is, assuming that relational variables range over sets that are either finite or are "coming into being," what definitions can be admitted as similarly legitimate? Surely, such definitions include explicit projections ($\{x \mid R(\vec{t})\}$, where R is a relation variable), as well as definitions by finite unions and intersections. For instance, if A and B are "evolving" infinite sets, then it is easy to describe a process that generates the set $A \cup B$.

The use of unrestricted projection (i.e., existential quantification) is also plausible in defining new predicative relations, though perhaps less so than the finitistic operations: if $R(x, y)$ is a generated relation, then $\{x \mid \exists y R(x, y)\}$ can be generated effectively in tandem with the generation of R. However, the universal quantifier is highly suspect, because it refers to an exhaustive inspection of the structure universe. Most importantly, negation and implication must be rejected, because accepting as legitimate the complement of an infinite set S is tantamount to stipulating that S is a completed totality.

The acceptance of infinite sets as evolving rather than completed totalities corresponds, therefore, to set-existence for at least the positive quantifier-free formulas (i.e., without negation or implication), and at most the positive existential formulas. Theorem 4.3 shows that these lower and upper bounds on predicativity yield the same class of provable functions, namely poly-time.

This reading of Theorem 4.3 suggests, then, that $\mathrm{IL}_2(\Sigma_1^+)$ is a natural formalism for feasible mathematics.[5] Formalisms for feasible arithmetic have been developed rather extensively following the seminal work of S. Buss [Bus86].[6] However, the use of weak second-order logic proposed here offers several advantages. For one, it is generic with respect to the data objects: although we focus on the algebra $\mathbb{W}$ (because Turing machine computation over $\{0, 1\}^*$ is the yardstick of computational complexity), weak second-order logics are not committed to any choice of data objects.

[5] One would opt, of course, for the strongest set existence principle among those yielding poly-time. The use of classical second-order logic in feasible mathematics is justified by the extension of Theorem 4.3, which we prove elsewhere.

[6] Also, a logical characterization of poly-time, based on linear logic, was developed in [GSS92].

Related to this generality is the absence, in our treatment, of any initial functions and of axioms for them.

The weak forms of set-existence considered here can be combined with the machinery of bounded arithmetic. Such formalisms have been considered by Ignjatovic, leading to an elegant proof theoretic characterization of the poly-time hierarchy [Ign92].

5. Every Poly-time Function Is w-Provable

5.1. *Provability of the Multiplicative String Function*

LEMMA 5.1. *The function* $\odot$ *is provable in* $\mathrm{ML}_2(QF^+)$.

Proof. Let P be the program in Section 2.2 for $\odot$. We derive in $\mathrm{ML}_2(QF^+)$ the formula

$$\tilde{\forall}P \wedge W[x, y] \to W[x \odot y].$$

Assume $\tilde{\forall}P$, $W[x, y]$, and $Cl[Q]$. We need to derive $Q(x \odot y)$. First, we show

$$Q(t) \to Q(x \oplus t). \tag{1}$$

From $\tilde{\forall}P$, using $Cl[Q]$, we have $\forall u\, Q(u \oplus t) \to Q(\mathbf{c}u \oplus t)$ $(\mathbf{c} = \mathbf{0}, \mathbf{1})$, and from $Q(t)$ we get $Q(\varepsilon \oplus t)$. By $W[x]$, with set-existence for the atomic formula $\varphi[u] \equiv Q(u \oplus t)$, the latter two formulas imply $Q(x \oplus t)$.

Putting $t \equiv x \odot z$ in (1), and using P, we get

$$Q(x \odot z) \to Q(x \odot \mathbf{c}z). \tag{2}$$

Since $x \odot \varepsilon = \varepsilon$, and $Q(\varepsilon)$ is given (from the assumption $Cl[Q]$), we also have

$$Q(x \odot \varepsilon). \tag{3}$$

By $W[y]$, with set-existence for the atomic formula $\varphi[u] \equiv Q(x \odot u)$, (2) and (3) imply $Q(x \odot y)$, concluding the proof. ∎

Note that the proof that $\odot$ is provable is somewhat different from the proof that $\oplus$ is provable. For $\oplus$ we proved $Cl[Q] \wedge Q(x) \wedge W[y] \to Q(x \oplus y)$, whereas for $\odot$ we proved $Cl[Q] \wedge W[x] \wedge W[y] \to Q(x \odot y)$. This difference explains why the process cannot be repeated to yield the provability of exponentiation. The difference between "local" uses of input (as for the first argument of $\oplus$) and "global" uses (as for the second argument) is an implicit manifestation of a concept dubbed "tiering" in [Lei90b], where it is used to obtain a subrecursive characterization of the

functions representable in the simply typed λ-calculus. A similar construct was discovered independently in [BC92] (triggered by the presentation in [Lei91] of the proof above), to obtain a subrecursive characterization of poly-time. That work was extended in [Blo92] to other complexity classes. Related results and refinements are in [Lei93].

5.2. *W-Provability of Poly-time Functions*

LEMMA 5.2. *Let* M, $P = P_M$, $\mathbf{r}$, $\boldsymbol{l}$, *and* $\mathbf{st}$ *be as in Lemma* 2.2. *Let*

$$\kappa[u] \underset{\text{df}}{\equiv} Q(\mathbf{r}(w, u)) \wedge Q(\boldsymbol{l}(w, u)) \wedge \text{State}[\mathbf{st}(w, u)],$$

where

$$\text{State}[s] \underset{\text{df}}{\equiv} \bigvee_{0 \leqslant i \leqslant n} \mathbf{s} = \mathbf{0}^{n-i}\mathbf{1}^i\boldsymbol{\varepsilon}.$$

Then

$$\tilde{\forall}P, Cl^+[Q], Cl^-[Q] \vdash_{\text{ML}_2(QF^+)} \kappa[t] \to \kappa[\mathbf{c}t] \qquad \textit{for} \quad \mathbf{c} = \mathbf{0}, \mathbf{1}.$$

Proof. By $\tilde{\forall}P$ we have $\mathbf{r}(w, \mathbf{c}z) = \boldsymbol{r}_1(\mathbf{st}(w, z), \boldsymbol{l}(w, z), \mathbf{r}(w, z))$. Assuming $Cl^-[Q]$, $Q(\mathbf{r}(w, z))$ implies that $\mathbf{r}(w, z)$ is $\boldsymbol{\varepsilon}$ or $\mathbf{cpr}(w, z)$ (with $\mathbf{c} = \mathbf{0}$ or $\mathbf{1}$), and similarly for $\boldsymbol{l}(w, z)$. Since State$[\mathbf{st}(w, t)]$, it follows that, in any event, the arguments of $\boldsymbol{r}_1$, i.e., $\mathbf{st}(w, z)$, $\boldsymbol{l}(w, z)$, $\mathbf{r}(w, z)$, fall under one of the defining equations for $\boldsymbol{r}_1$.

The value of $\mathbf{r}(w, \mathbf{c}t)$, obtained by the clause for $\mathbf{st}(w, z)$ (which is unique since the machine is deterministic), is either $\mathbf{r}(w, z)$, $\mathbf{0r}(w, z)$, $\mathbf{1r}(w, z)$, or $\mathbf{pr}(w, z)$. By $Cl^+[Q]$ and $Cl^-[Q]$, we have $Q(\mathbf{r}(w, \mathbf{c}z))$ in either one of these cases.

Similarly, we obtain $Q(\boldsymbol{l}(w, \mathbf{c}z))$ and State$[\mathbf{st}(w, \mathbf{c}z)]$. ∎

LEMMA 5.3. *Let* M, P, $\mathbf{r}$, $\boldsymbol{l}$, $\mathbf{st}$, *and* κ *be as above. Then*

$$\tilde{\forall}P \vdash_{\text{ML}_2(QF^+)} W[w, t] \to \widehat{W}[\mathbf{r}(w, t)].$$

Proof. We derive in $\text{ML}_2(QF^+)$

$$\tilde{\forall}P, W[t], W[w], \widehat{Cl}[Q] \Rightarrow \kappa[t]. \tag{*}$$

Since $Q(\mathbf{r}(w, t))$ is a conjunct of $\kappa[t]$, this implication yields

$$\tilde{\forall}P \wedge W[w, t] \to \forall Q(\widehat{Cl}[Q] \to Q(\mathbf{r}(w, t)),$$

proving the theorem.

Assume the premises of (*). From $W[t]$ we have $Cl[\kappa] \to \kappa[t]$, by set-existence for the positive quantifier-free formula κ. We need only prove $Cl[\kappa]$.

• $\kappa[\varepsilon]$: From $\widehat{Cl}[Q]$ and $W[w]$ we have $Q(w)$. By $\tilde{\forall}P$, $\mathbf{r}(w, \varepsilon) = w$, and so $Q(\mathbf{r}(w, \varepsilon))$. From $\widehat{Cl}[Q]$ we have $Q(\varepsilon)$. But $\tilde{\forall}P$ implies $\boldsymbol{l}(w, \varepsilon) = \varepsilon$, so $Q(\boldsymbol{l}(w, \varepsilon))$. State$[\mathbf{st}(w, \varepsilon)]$ is immediate.

• $\kappa[z] \to \kappa[\mathbf{c}(z)]$ is Lemma 5.2. ∎

Proof of Lemma 4.4. Let M be a deterministic TM computing a function f, which for input w runs in time $\leqslant m \cdot |w|^k$ for some m, k. Let $(P_M, \mathbf{r})$ be defined as above. By Lemma 5.3 the function r computed by $(P_M, \mathbf{r})$ is w-provable in $\mathrm{ML}_2(QF^+)$. By Lemma 5.1 the function $T(w) =_{\mathrm{dfg}} \mathbf{0}^m \boldsymbol{\varepsilon} \odot w \odot \cdots \odot w$ (k factors) is provable in $\mathrm{ML}_2(QF^+)$ (since provability is closed under composition). Since $f(w) = r(w, T(w))$, and since the composition of a provable function with a w-provable function is w-provable, it follows from Lemma 5.1 that f is w-provable. ∎

6. Provable Functions Are Poly-time

6.1. *A Natural Deduction Formalism for Second-Order Logic*

We use a proof theoretic analysis, referring to the following variant of Gentzen–Prawitz's natural-deduction formalism for intuitionistic second-order logic [Pra65]. Though equality is second-order definable, we use it as a primitive, because we wish to consider equations as atomic formulas. Negation is defined in terms of the 0-ary logical constant $\bot$, denoting falsehood: $\neg\varphi \equiv_{\mathrm{df}} (\varphi \to \bot)$.

A *sequent* is a pair $\Gamma \Rightarrow \varphi$, where Γ is a finite set of formulas and φ is a formula. We abbreviate $\Gamma \cup \{\psi\}$ by Γ, ψ. Each *inference rule* is a schema (i.e., template) for deriving a sequent (the *conclusion*) from other sequents (the *premises*). Each rule has 0, 1, 2, or 3 premises. The rules are as follows. We use the customary labels: init, intro, elim, and abs for initial, introduction, elimination, and absurdum, respectively.

Init. $\Gamma \Rightarrow \varphi$, where $\varphi \in \Gamma$ (no premise).

=-intro. $\Gamma \Rightarrow \mathbf{t} = \mathbf{t}$ (no premise).

=-elim. $\Gamma \Rightarrow [\mathbf{t}/u]\varphi$ is derived from $\Gamma \Rightarrow [\mathbf{t}'/u]\varphi$ and $\Gamma \Rightarrow \mathbf{t} = \mathbf{t}'$, where φ is atomic.

∧-intro. $\Gamma \Rightarrow \varphi \wedge \psi$ is derived from $\Gamma \Rightarrow \varphi$ and $\Gamma \Rightarrow \psi$.

∧-elim. $\Gamma \Rightarrow \varphi$ is derived from $\Gamma \Rightarrow \varphi \wedge \psi$. Also, $\Gamma \Rightarrow \psi$ is derived from $\Gamma \Rightarrow \varphi \wedge \psi$.

$\vee$-intro. $\Gamma \Rightarrow \varphi \vee \psi$ is derived from $\Gamma \Rightarrow \varphi$. Also, $\Gamma \Rightarrow \varphi \vee \psi$ is derived from $\Gamma \Rightarrow \psi$.

$\vee$-elim. $\Gamma \Rightarrow \varphi$ is derived from the three sequents $\Gamma \Rightarrow \psi_0 \vee \psi_1$, $\Gamma, \psi_0 \Rightarrow \varphi$, and $\Gamma, \psi_1 \Rightarrow \varphi$.

$\rightarrow$-intro. $\Gamma \Rightarrow \psi \rightarrow \varphi$ is derived from $\Gamma, \psi \Rightarrow \varphi$.

$\rightarrow$-elim. $\Gamma \Rightarrow \varphi$ is derived from $\Gamma \Rightarrow \psi \rightarrow \varphi$ and $\Gamma \Rightarrow \psi$.

$\exists$-intro. $\Gamma \Rightarrow \exists u \varphi$ is derived from $\Gamma \Rightarrow [\mathbf{t}/u]\varphi$ (where $[\mathbf{t}/u]\varphi$ is a correct substitution, i.e., no variable free in φ is being captured).

$\exists$-elim. $\Gamma \Rightarrow \varphi$ is derived from $\Gamma \Rightarrow \exists u \psi$ and $\Gamma, [z/u]\psi \Rightarrow \varphi$, provided the variable z is not free in Γ, φ.

$\forall$-intro. $\Gamma \Rightarrow \forall u \varphi$ is derived from $\Gamma \Rightarrow [z/u]\varphi$, provided z is not free in Γ.

$\forall$-elim. $\Gamma \Rightarrow [\mathbf{t}/u]\varphi$ is derived from $\Gamma \Rightarrow \forall u \varphi$ (where $[\mathbf{t}/u]\varphi$ is a correct substitution).

$\exists^2$-intro. $\Gamma \Rightarrow \exists R \varphi$ is derived from $\Gamma \Rightarrow [\lambda \vec{x}.\chi/R]\varphi$, where $[\lambda \vec{x}.\chi/R]\varphi$ denotes the result of replacing every subformula $R(\vec{\mathbf{t}})$ of φ by $[\vec{\mathbf{t}}/\vec{x}]\chi$. The formula χ is the *eigen-formula* of the inference.

$\exists^2$-elim. $\Gamma \Rightarrow \varphi$ is derived from $\Gamma \Rightarrow \exists R \psi$ and $\Gamma, [Q/R]\psi \Rightarrow \varphi$, where $\text{arity}(Q) = \text{arity}(R)$, and provided the variable Q is not free in Γ, φ.

$\forall^2$-intro. $\Gamma \Rightarrow \forall R \varphi$ is derived from $\Gamma \Rightarrow [Q/R]\varphi$ (where $\text{arity}(Q) = \text{arity}(R)$), provided Q is not free in Γ.

$\forall^2$-elim. $\Gamma \Rightarrow [\lambda \vec{x}.\chi/R]\varphi$ is derived from $\Gamma \Rightarrow \forall R \varphi$. The formula χ is the *eigen-formula* of the inference.

abs. $\Gamma \Rightarrow \varphi$ is derived from $\Gamma \Rightarrow \bot$, where φ is atomic.

A *derivation* is a list of sequents where each sequent follows from previously listed ones by one of the inference rules. A sequent $\Gamma \Rightarrow \varphi$ is derivable if it is the last in a derivation. We write $\Gamma \vdash_{\text{IL}_2} \varphi$ if the sequent $\Gamma \Rightarrow \varphi$ is derived. Let Φ be a collection of formulas. If $\Gamma \Rightarrow \varphi$ has a derivation in IL_2 where each eigen-formula is in Φ, then we say that $\Gamma \Rightarrow \varphi$ is derived in $\text{IL}_2(\Phi)$.

A natural-deduction calculus for *first*-order intuitionistic logic, IL_1, is obtained by restricting the rules above to first-order formulas; in particular, the second-order rules are not used.

6.2. *Normal Derivations and the Subformula Property*

For each one of the elimination rules above, the first premise is called the *major premise*, the remaining are the *minor premises*. The introduction rules, $\vee$-elim, $\exists$-elim, and $\exists^2$-elim, are called *pivotal*. If the major premise of an elimination rule is derived by a pivotal rule then it is called a *critical sequent*. A derivation is *normal* if it contains no critical sequents.

The following theorem is essentially due to Girard [Gir71, Pra71].

THEOREM 6.1 [Normal Form]. *If a sequent has a derivation in* IL_2, *then it has a normal derivation in* IL_2.

A subformula ψ of a first-order formula φ is a *positive* [*negative*] subformula of φ if it is in the negative (i.e., left-side) scope of an even [odd, respectively] number of implications. (Recall that negation is here a special kind of implication.) ψ is a *strictly positive* subformula of φ if it is not in the negative scope of any implication in φ. ψ is a *positive subformula* of a sequent $\Gamma \Rightarrow \varphi$ if it is a positive subformula of φ, or a negative subformula of an element of Γ.[7]

LEMMA 6.2 [Subformula]. *Suppose that* Π *is a normal derivation of* IL_1. *If a sequent* $\Gamma \Rightarrow \varphi$ *in* Π *is the major premise of an elimination rule, then* φ *is a positive subformula of* Γ.

Proof. By induction on Π. Since Π is normal, $\Gamma \Rightarrow \varphi$ cannot be derived by an introduction rule, by $\vee$-elimination, or by $\exists$-elim. This leaves us with two possibilities. Either $\Gamma \Rightarrow \varphi$ is initial, and then $\varphi \in \Gamma$, so φ is a positive subformula of itself and therefore of Γ. Or else $\Gamma \Rightarrow \varphi$ is derived by $\wedge$-elimination, $\rightarrow$-elimination, or $\forall$-elimination. If $\Gamma \Rightarrow \psi$ is the major premise of that inference, then φ is a positive subformula of ψ, and ψ is a positive subformula of Γ, by induction assumption. Thus φ is a positive subformula of Γ. ∎

LEMMA 6.3. *Let* Φ *be a class of first-order formulas. Suppose that* Π *is a normal* $\mathrm{IL}_2(\Phi)$ *derivation of a sequent of the form* $\Gamma, \forall R\psi \Rightarrow \varphi$, *where* Γ, ψ *are all first-order. If* φ *contains a second-order quantifier, then the last inference of* Π *is either pivotal or* [*init*].

Proof. By induction on Π. Towards a contradiction, assume that φ contains a second-order quantifier and is derived by an elimination rule other than $\vee$-elim, $\exists$-elim, and $\exists^2$-elim, with major premise $\Gamma, \forall R\psi \Rightarrow \varphi'$. Then φ' must contain a second-order quantifier, and so, by induction assumption, the sequent $\Gamma, \forall R\psi \Rightarrow \varphi'$ is derived by either a pivotal rule or [init]. Pivotal rules are excluded, since Π is normal. If the inference is [init], then $\varphi' \equiv \forall R\psi$, and so $\varphi \equiv [\lambda \vec{u}\chi/R]\psi$ with $\chi \in \Phi$, so φ is first-order, contradicting the lemma's assumptions. ∎

[7] We refer to the common meaning of "subformula," by which each formula $[\mathbf{t}/u]\psi$ is a subformula of $\forall u\psi$.

6.3. *A Reduction to First-Order Proofs*

In the remainder of this section we show that every function w-provable in $\mathrm{IL}_2(\Sigma_1^+)$ is poly-time. Our proof is in two steps. First, we show that from a proof in $\mathrm{IL}_2(\Sigma_1^+)$ of a formula $\widetilde{\forall} P \to \widehat{W}[x] \to W[\mathbf{f}(x)]$ we can extract a natural deduction derivation of IL_1 (with a useful additional property) that proves a sequent expressing the w-provability of $\mathbf{f}^P$. Then, from the latter deduction we extract a poly-time algorithm for $\mathbf{f}^P$. In this subsection we establish the first reduction.

LEMMA 6.4. *Let Φ be a collection of first-order formulas. Suppose that Π is a normal derivation for a sequent of the form $\Gamma, W[x] \Rightarrow \varphi$, where Γ and φ are first order. Then* IL_1 *derives a sequent of the form $\Gamma, \{Cl[\alpha_i] \to \alpha_i[x]\}_{i<I} \Rightarrow \varphi$, with $\alpha_i \in \Phi$ $(i \in I)$.*

Proof. By induction on Π, considering cases for the last inference. If that inference is $\wedge$-intro, with premises $\Gamma, W[x] \Rightarrow \varphi_1$ and $\Gamma, W[x] \Rightarrow \varphi_2$, then, by induction assumption, $\Gamma, \{Cl[\alpha_i] \to \alpha_i[x]\}_{i<I} \vdash_{\mathrm{IL}_1} \varphi_1$ and $\Gamma, \{Cl[\alpha_j'] \to \alpha_j'[x]\}_{j<J} \vdash_{\mathrm{IL}_1} \varphi_2$, for some formulas $\alpha_i, \alpha_j' \in \Phi$. Therefore $\Gamma, \{Cl[\alpha_i] \to \alpha_i[x]\}_{i<I}, \{Cl[\alpha_j'] \to \alpha_j'[x]\}_{j<J} \vdash_{\mathrm{IL}_1} \varphi$, by $\wedge$-intro. The cases for Init, $=$-intro, $=$-elim, $\vee$-intro, $\to$-intro, $\exists$-intro, $\forall$-intro, and $\forall$-elim are either trivial or similar. The rules $\forall^2$-intro and $\exists^2$-intro are excluded since φ is first-order.

If the inference is $\wedge$-elim, and φ is derived from $\varphi \wedge \psi$, then $\varphi \wedge \psi$ is a subformula of the derived sequent, by Lemma 6.2. It follows that $\varphi \wedge \psi$ is also first order, so induction assumption can be used as above. The case for $\to$-elim, $\vee$-elim, $\exists$-elim, and $\exists^2$-elim are similar.

If the inference is $\forall^2$-elim, then, by Lemma 6.3, the premise must be $W[x]$, and so φ must be of the form $Cl[\alpha] \to \alpha[x]$, where α is first order. The lemma's statement is then immediate. ∎

6.4. *A Notion of Realizability Based on Equational Computability.*

Our aim is to extract poly-time constructive contents from derivations of sequents of the form $\widetilde{\forall} P, \{Cl[\alpha_i] \to \alpha_i[x]\}_{i<I}, \widehat{Cl}[Q] \Rightarrow Q(\mathbf{f}(x))$ $(\alpha_i \in \Sigma_1^+)$. We do this using a variant of Kleene's notion of constructive realization of formulas (see, e.g., [Tro73]). Fix a coherent equational program P over $\mathbb{W}$, let V_P be (as before) the vocabulary of P and $\mathbb{W}$, and let $\Sigma_1^+(P, Q)$ be the set of positive existential formulas over the vocabulary V_P augmented with the unary relation variable Q. Let T be the set of closed V_P-terms, and let R be the set of lists with the elements of T as atoms. We identify a singleton list $\langle \mathbf{t} \rangle$ with its sole entry $\mathbf{t}$. We write $\langle a_0, a_1 \rangle$ for the pair of a_0 and a_1, and proj_i $(i = 0, 1)$ for the ith projection.

For closed formulas $\varphi \in \Sigma_1^+(P, Q)$ and $\rho \in R$ we define the relation $\rho \Vdash_P \varphi$ by recurrence on φ, as follows. We let $\bar{0} =_{\mathrm{df}} \boldsymbol{\varepsilon}$, $\bar{1} =_{\mathrm{df}} \mathbf{1}\boldsymbol{\varepsilon}$.

$$\begin{array}{lcl}
\rho \Vdash_P \bot & & \text{for no } \rho \\
\rho \Vdash_P \mathbf{t} = \mathbf{t}' & \underset{\mathrm{df}}{\equiv} & \rho = \boldsymbol{\varepsilon} \text{ and } P \vdash \mathbf{t} = \mathbf{t}' \\
\rho \Vdash_P Q(\mathbf{t}) & \underset{\mathrm{df}}{\equiv} & \rho \in \mathbb{W} \text{ and } P \vdash \mathbf{t} = \rho \\
\rho \Vdash_P \varphi_0 \wedge \varphi_1 & \underset{\mathrm{df}}{\equiv} & \rho = \langle \rho_0, \rho_1 \rangle \text{ where } \rho_i \Vdash_P \varphi_i \\
\rho \Vdash_P \varphi_0 \vee \varphi_1 & \underset{\mathrm{df}}{\equiv} & \rho = \langle \bar{\imath}, \rho' \rangle \text{ where } \rho' \Vdash_P \varphi_i \\
\rho \Vdash_P \exists u \varphi & \underset{\mathrm{df}}{\equiv} & \rho = \langle \mathbf{t}, \rho' \rangle \text{ where } \rho' \Vdash_P [\mathbf{t}/u]\varphi \text{ and } \mathbf{t} \in T.
\end{array}$$

6.5. *Provable Functions Are Poly-time*

We show that a normal derivation Δ of

$$\widetilde{\forall} P, \{Cl[\alpha_i] \to \alpha_i[x]\}_{i \in I}, \widehat{Cl}[Q] \Rightarrow Q(\mathbf{f}(x))$$

yields a poly-time realizability for certain first-order sequents in Δ, and in particular for the derived sequent. The computation model we use for these realizing functions is a symbolic register machine, where the values stored in registers are closed terms, and where copying the contents of one register to another counts as a single computation step. Clearly, a computation in this model which runs in polynomial time can be simulated by a (multi-tape) TM running in polynomial time. However, there may be an increase in the degree of the polynomials when passing to the simulating machine; for example, copying takes constant time on the symbolic register machine, but linear time on a TM.

LEMMA 6.5. [Realizability]. *Let P be an equational program that contains the defining equations for* $\mathbf{p}$. *Suppose that Δ is a derivation of a sequent of the form $\Gamma \Rightarrow \varphi$, where*

$$\Gamma = \widetilde{\forall} P, \{Cl[\alpha_i] \to \alpha_i[x]\}_{i \in I}, \beta_1, \ldots, \beta_n, Q[\boldsymbol{\varepsilon}], Cl^+[Q], Cl^-[Q],$$

where $\varphi, \alpha_i, \beta_j \in \Sigma_1^+$. Let $\vec{u} = u_1 \cdots u_k$ be a listing of the variables free in $\vec{\beta}, \varphi$, other than x.

Then there is a function $h: \mathbb{W} \times T^k \times R^n \to R$, computable by a symbolic register machine, which is poly-time in its first argument and constant time in its remaining arguments, and such that, for all $\vec{\mathbf{s}} \in T^k$ and $\rho_1 \cdots \rho_n \in R$, if

$$\rho_j \Vdash_P \beta_j[\mathbf{w}, \vec{\mathbf{s}}] \quad \text{for } j = 1 \cdots n, \qquad \text{then} \quad h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_P \varphi[\mathbf{w}, \vec{\mathbf{s}}]$$

(*where we write $\psi[\mathbf{w}, \vec{\mathbf{s}}]$ for $[\mathbf{w}, \vec{\mathbf{s}}/x, \vec{u}]\psi$*).

We postpone the proof to the next subsection.

Proof of Lemma 4.5. The premise of the lemma implies that

$$\tilde{\forall} P,\ \{Cl[\alpha_i] \to \alpha_i[x]\}_{i \in I},\ \widehat{Cl}[Q] \vdash_{\mathrm{IL}_1} Q(\mathbf{f}(x)),$$

where $\alpha_i \in \Sigma_1^+(Q)$. It follows, by the Realizability Lemma, that there exists a poly-time function h such that, for all $\mathbf{w} \in \mathbb{W}$, $h(\mathbf{w}) \Vdash_{\bar{P}} Q(\mathbf{f}(\mathbf{w}))$. Thus $f(\mathbf{w}) = h(\mathbf{w}) \in \mathbb{W}$, and so f is poly-time. ∎

6.6. *Proof of the Realizability Lemma*

Let Δ be a normal derivation of $\Gamma \Rightarrow \varphi$. The proof of the lemma is by induction on Δ, considering cases for the last inference of Δ.

We write $\vec{\rho} \Vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$ for the (metamathematical) conjunction of all statements $\rho_j \Vdash_{\bar{P}} \beta_j[\mathbf{w}, \vec{\mathbf{s}}]$, $j = 1 \cdots n$.

Init. Since $\varphi \in \Sigma_1^+$, φ is either $Q(\boldsymbol{\varepsilon})$ or a β_j. In the first case let $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) =_{\mathrm{df}} \boldsymbol{\varepsilon}$, and in the latter case let $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) =_{\mathrm{df}} \rho_j$.

=-intro. φ is $\mathbf{t} = \mathbf{t}$. Let $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) =_{\mathrm{df}} \boldsymbol{\varepsilon}$.

=-elim. $\varphi \equiv [\mathbf{t}/u]\psi$ is derived from $[\mathbf{t}'/u]\psi$ and $\mathbf{t} = \mathbf{t}'$. Without loss of generality, u is not among $u_1 \cdots u_k$; write u_{k+1} for u. By induction assumption there are functions h_0, h_1, of the required time-complexity, with the following properties. If $\vec{\rho} \Vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_{\bar{P}} ([\mathbf{t}'/u]\psi)[\mathbf{w}, \vec{\mathbf{s}}]$ and $h_1(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_{\bar{P}} (\mathbf{t} = \mathbf{t}')[\mathbf{w}, \vec{\mathbf{s}}]$.

Let $h = h_0$. Since the $=$-elim rule applies only to atomic formulas, the formula ψ is either of the form $Q(\mathbf{r})$ or is an equation. Suppose ψ is $Q(\mathbf{r})$, and assume $\vec{\rho} \Vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$. Then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_{\bar{P}} ([\mathbf{t}'/u]\, Q(\mathbf{r}))[\mathbf{w}, \vec{\mathbf{s}}]$, that is (by the definition of realizability for atomic formulas)

$$P \vdash \mathbf{r}[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{t}'[\mathbf{w}, \vec{\mathbf{s}}]] = h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}),$$

with $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \in \mathbb{W}$. Also, $\vec{\rho} \Vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$ implies $h_1(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_{\bar{P}} (\mathbf{t} = \mathbf{t}')[\mathbf{w}, \vec{\mathbf{s}}]$, whence $P \vdash \mathbf{t}[\mathbf{w}, \vec{\mathbf{s}}] = \mathbf{t}'[\mathbf{w}, \vec{\mathbf{s}}]$. Combining this with our former conclusion we find that

$$P \vdash \mathbf{r}[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{t}[\mathbf{w}, \vec{\mathbf{s}}]] = h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}),$$

i.e. $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_{\bar{P}} ([\mathbf{t}/u]\, Q(\mathbf{r}))[\mathbf{w}, \vec{\mathbf{s}}]$.

The proof for the case where ψ is an equation is similar.

∧-intro. $\varphi \equiv \varphi_0 \wedge \varphi_1$ is derived from φ_0 and φ_1. By induction assumption there are functions h_0, h_1, with the required time-complexity, and such that if $\vec{\rho} \Vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_i(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_{\bar{P}} \varphi_i[\mathbf{w}, \vec{\mathbf{s}}]$ $(i = 0, 1)$. Let

$$h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \underset{\mathrm{df}}{=} \langle h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}), h_1(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \rangle.$$

Since our symbolic register machines copy in constant time, h has the required time complexity. And since $h_i(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \varphi_i[\mathbf{w}, \vec{\mathbf{s}}]$ $(i = 0, 1)$, we also have $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \varphi[\mathbf{w}, \vec{\mathbf{s}}]$.

$\wedge$-elim. φ is derived from, say, $\varphi \wedge \psi$. By the Subformula Lemma, $\varphi \wedge \psi$ is a positive subformula of Γ. For all such subformulas, if $\varphi \in \Sigma_1^+$ then $(\varphi \wedge \psi) \in \Sigma_1^+$. Therefore induction assumption applies. Let $u_{k+1} \cdots u_{k+\ell}$ be the variables free in ψ that are not among $x, u_1 \cdots u_k$. By induction assumption there is a poly-time h_0 such that, if $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}, \boldsymbol{\varepsilon}']$, i.e. (since $u_{k+1} \cdots u_{k+\ell}$ are not free in $\vec{\beta}$) if $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \boldsymbol{\varepsilon}', \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} (\varphi \wedge \psi)[\mathbf{w}, \vec{\mathbf{s}}, \boldsymbol{\varepsilon}']$. Let

$$h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \underset{\text{df}}{=} \text{proj}_0(h_0(\mathbf{w}, \vec{\mathbf{s}}, \boldsymbol{\varepsilon}', \vec{\rho})).$$

The case for the dual form of $\wedge$-elimination is analogous.

$\vee$-intro. $\varphi \equiv \varphi_0 \vee \varphi_1$ is derived from, say, φ_0. By induction assumption there is a function h_0, of the required time-complexity, such that, if $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \varphi_0[\mathbf{w}, \vec{\mathbf{s}}]$. Let

$$h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \underset{\text{df}}{=} \langle \bar{0}, h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \rangle.$$

Now, if $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$ then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \varphi_0[\mathbf{w}, \vec{\mathbf{s}}]$, whence $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \varphi[\mathbf{w}, \vec{\mathbf{s}}]$. The case for the dual form of $\vee$-introduction is analogous.

$\vee$-elim. $\Gamma \Rightarrow \varphi$ is derived from sequents of the forms $\Gamma \Rightarrow \psi_0 \vee \psi_1$, $\Gamma, \psi_0 \Rightarrow \varphi$, and $\Gamma, \psi_1 \Rightarrow \varphi$. Then, by the Subformula Lemma, $\psi_0 \vee \psi_1$ is a positive subformula of Γ, i.e. is either a subformula of $\mathbf{t} = \boldsymbol{\varepsilon} \vee \exists u(Q(u) \wedge \mathbf{t} = \mathbf{0}u) \vee \exists u(Q(u) \wedge \mathbf{t} = \mathbf{1}u)$ for some $\mathbf{t}$, or is a subformula of some α_i or β_j. In either case $(\psi_0 \vee \psi_1) \in \Sigma_1^+$, and by the induction assumption there are functions $h_\vee$, h_0, and h_1 of the required time-complexity, with the following properties.

1. If $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_\vee(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} (\psi_0 \vee \psi_1)[\mathbf{w}, \vec{\mathbf{s}}]$, i.e., $h_\vee(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) = \langle \bar{i}, \rho' \rangle$, where $i = 0$ or 1, and $\rho' \vdash\!\!\!\vdash_{\bar{P}} \psi_i[\mathbf{w}, \vec{\mathbf{s}}]$.

2. For $i = 0, 1$: If $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, and if $\tau \vdash\!\!\!\vdash_{\bar{P}} \psi_i[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_i(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}, \tau) \vdash\!\!\!\vdash_{\bar{P}} \varphi[\mathbf{w}, \vec{\mathbf{s}}]$.

Let $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) =_{\text{df}}$ if $h_\vee(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})$ is $\langle \bar{i}, \rho' \rangle$ then $h_i(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}, \rho')$.

Assume $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$; then $h_\vee(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} (\psi_0 \vee \psi_1)[\mathbf{w}, \vec{\mathbf{s}}]$, i.e., $h_\vee(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) = \langle \bar{i}, \rho' \rangle$ where $\rho' \vdash\!\!\!\vdash_{\bar{P}} \psi_i[\mathbf{w}, \vec{\mathbf{s}}]$. So $h_i(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}, \rho') \vdash\!\!\!\vdash_{\bar{P}} \varphi[\mathbf{w}, \vec{\mathbf{s}}]$; i.e., $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \varphi[\mathbf{w}, \vec{\mathbf{s}}]$.

$\rightarrow$-intro. This is excluded, since $\varphi \in \Sigma_1^+$.

$\rightarrow$-elim. φ is derived from $\psi \rightarrow \varphi$ and ψ. Since Δ is normal, the premise $\psi \rightarrow \varphi$ cannot be derived by an introduction. By the Subformula Lemma it follows that $\psi \rightarrow \varphi$ is a positive subformula of Γ.

Subcase I. $\psi \in \Sigma_1^+$. By induction assumption there is a function h_0 such that, if $\vec{\rho} \Vdash_P \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_P \psi[\mathbf{w}, \vec{\mathbf{s}}]$. We have the following possibilities for ψ.

1. $\psi \to \varphi$ is a subformula of $Cl^+[Q]$, i.e., for some term $\mathbf{t}$, ψ is $Q(\mathbf{t})$ and φ is $Q(\mathbf{ct})$, where $\mathbf{c} = \mathbf{0}$ or $\mathbf{1}$. Let

$$h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \underset{\text{df}}{=} \mathbf{c}(h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})).$$

2. $\psi \to \varphi$ is a subformula of $Cl\ [Q]$, i.e., for some term $\mathbf{t}$, ψ is $Q(\mathbf{t})$ and φ is $Bk[Q, \mathbf{t}]$. Let $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) =_{\text{df}}$ if $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})$ is $\boldsymbol{\varepsilon}$ then $\langle \bar{0}, \boldsymbol{\varepsilon} \rangle$ else if $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})$ is $\mathbf{0v}$ for some $\mathbf{v}$ then $\langle \bar{1}, \langle \bar{0}, \langle \mathbf{v}, \langle \mathbf{v}, \boldsymbol{\varepsilon} \rangle \rangle \rangle$ else if $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})$ is $\mathbf{1v}$ for some $\mathbf{v}$ then $\langle \bar{1}, \langle \bar{1}, \langle \mathbf{v}, \langle \mathbf{v}, \boldsymbol{\varepsilon} \rangle \rangle \rangle$.

Now, if $\vec{\rho} \Vdash_P \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_P Q(\mathbf{t})[\mathbf{w}, \vec{\mathbf{s}}]$, i.e., $P \vdash \mathbf{t}[\mathbf{w}, \vec{\mathbf{s}}] = h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})$, where $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \in \mathbb{W}$. If $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) = \mathbf{t}[\mathbf{w}, \vec{\mathbf{s}}] = \boldsymbol{\varepsilon}$, then $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) = \langle \bar{0}, \boldsymbol{\varepsilon} \rangle$, and so $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_P (\mathbf{t} = \boldsymbol{\varepsilon} \vee \chi)[\mathbf{w}, \vec{\mathbf{s}}]$ for any formula χ. If $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) = \mathbf{t}[\mathbf{w}, \vec{\mathbf{s}}] = \mathbf{0v}$ for some $\mathbf{v} \in \mathbb{W}$, then $P \vdash \mathbf{p}(\mathbf{t}[\mathbf{w}, \vec{\mathbf{s}}]) = \mathbf{v}$, since P is assumed to contain the defining equations for $\mathbf{p}$. Hence

$$\langle \mathbf{v}, \langle \mathbf{v}, \boldsymbol{\varepsilon} \rangle \rangle \Vdash_P \exists v(Q(v) \wedge \mathbf{t}[\mathbf{w}, \vec{\mathbf{s}}] = \mathbf{0}v),$$

and so

$$\langle \bar{0}, \langle \mathbf{v}, \langle \mathbf{v}, \boldsymbol{\varepsilon} \rangle \rangle \rangle \Vdash_P \exists v(Q(v) \wedge \mathbf{t} = \mathbf{0}v) \vee \exists v(Q(v) \wedge \mathbf{t} = \mathbf{1}v).$$

Therefore

$$\langle \bar{1}, \langle \bar{0}, \langle \mathbf{v}, \langle \mathbf{v}, \boldsymbol{\varepsilon} \rangle \rangle \rangle \rangle \Vdash_P Bk[Q, \mathbf{t}].$$

The case where $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})$ is of the form $\mathbf{1v}$ is analogous. In any event, $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_P \varphi[\mathbf{w}, \vec{\mathbf{s}}]$.

Subcase II. For some $\beta[u] \in \Sigma_1^+$, ψ is $Cl[\beta]$ and φ is $\beta[x]$. (This subcase is the heart of the proof.) The formula $Cl[\beta]$ is not a positive subformula of Γ, so it cannot be derived by an elimination rule, by the Subformula Lemma. Thus $Cl[\beta]$ is derived in Δ by successive introductions, from sequents of the form $\Gamma \Rightarrow \beta[\boldsymbol{\varepsilon}]$, $\Gamma \Rightarrow \beta[y] \to \beta[\mathbf{0}y]$, and $\Gamma \Rightarrow \beta[z] \to \beta[\mathbf{1}z]$, where y and z are not free in Γ. If one of the latter two sequents is derived by an elimination, then, by the Subformula Lemma, they are derived from $\widehat{Cl}[Q]$, thus $\beta[u]$ is $Q(u)$, and φ is $Q(x)$ (by the subcase condition). Then $\mathbf{w} \Vdash_P \varphi[\mathbf{w}, \vec{\mathbf{s}}]$, and we set $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) =_{\text{df}} \mathbf{w}$.

Otherwise, both $\beta[y] \to \beta[\mathbf{0}y]$ and $\beta[z] \to \beta[\mathbf{1}z]$ are derived by introductions. We then have sub-derivations Δ_ε, Δ_0, and Δ_1 of Δ, for the sequents $\Gamma \Rightarrow \beta[\boldsymbol{\varepsilon}]$, $\Gamma, \beta[y] \Rightarrow \beta[\mathbf{0}y]$, and $\Gamma, \beta[z] \Rightarrow \beta[\mathbf{1}z]$, respectively. By induction assumption applied to Δ_ε, Δ_0, and Δ_1, there are functions h_ε,

h_0, and h_1, of the required time-complexity, and with the following properties (we take y, respectively z, to be u_{k+1} in the extended list $\vec{u} = (u_1 \cdots u_{k+1})$ of free variables).

1. if $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_\varepsilon(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \beta[\mathbf{w}, \vec{\mathbf{s}}, \boldsymbol{\varepsilon}]$;
2. for all $\mathbf{y} \in \mathbb{W}$: if $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, and $\tau \vdash\!\!\!\vdash_{\bar{P}} \beta[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{y}]$, then

$$h_c(\mathbf{w}, \vec{\mathbf{s}}, \mathbf{y}, \vec{\rho}, \tau) \vdash\!\!\!\vdash_{\bar{P}} \beta[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{cy}] \qquad \mathbf{c} = \mathbf{0}, \mathbf{1}.$$

(Note that $\beta[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{y}]$ is $[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{y}/x, \vec{u}, u_{k+1}]\beta$, and that x may already be present in $\beta[u]$, so that $\varphi[\mathbf{w}]$ is $[\mathbf{w}/x]\beta$, i.e. $\beta[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{w}]$.)

Define h' by recurrence:

$$h'(\mathbf{w}, \boldsymbol{\varepsilon}, \vec{\mathbf{s}}, \vec{\rho}) = h_\varepsilon(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})$$

$$h'(\mathbf{w}, \mathbf{cy}, \vec{\mathbf{s}}, \vec{\rho}) = h_c(\mathbf{w}, \vec{\mathbf{s}}, \mathbf{y}, \vec{\rho}, h'(\mathbf{w}, \mathbf{y}, \vec{\mathbf{s}}, \vec{\rho})) \qquad \mathbf{c} = \mathbf{0}, \mathbf{1}$$

and let

$$h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \underset{\text{df}}{=} h'(\mathbf{w}, \mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}).$$

Since h_0 and h_1 are poly-time in their first argument and constant-time in their remaining arguments, it follows that h' is poly-time in its first argument, linear-time in the second, and constant-time in the remaining arguments. Therefore h is poly-time in its first argument (with the polynomial-degree 1 + the largest of the degrees of h_0 and h_1), and constant-time in the remaining arguments.

We show, by induction on $\mathbf{y} \in \mathbb{W}$, that if $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h'(\mathbf{w}, \mathbf{y}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \beta[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{y}]$. For $\mathbf{y} = \boldsymbol{\varepsilon}$ we have $h'(\mathbf{w}, \boldsymbol{\varepsilon}, \vec{\mathbf{s}}, \vec{\rho}) = h_\varepsilon(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})$, and so $h'(\mathbf{w}, \boldsymbol{\varepsilon}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \beta[\mathbf{w}, \vec{\mathbf{s}}, \boldsymbol{\varepsilon}]$. Assume $h'(\mathbf{w}, \mathbf{y}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \beta[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{y}]$. Then, for $c = 0, 1$, the property of h_c implies that

$$h_c(\mathbf{w}, \vec{\mathbf{s}}, \mathbf{y}, \vec{\rho}, h'(\mathbf{w}, \mathbf{y}, \vec{\mathbf{s}}, \vec{\rho})) \vdash\!\!\!\vdash_{\bar{P}} \beta[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{cy}],$$

i.e.,

$$h'(\mathbf{w}, \mathbf{cy}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \beta[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{cy}].$$

This concludes the induction. We therefore get $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \beta[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{w}]$, i.e.,

$$h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} \varphi[\mathbf{w}, \vec{\mathbf{s}}].$$

∃-intro. $\varphi \equiv \exists u \psi$ is derived from $[\mathbf{t}/u]\psi$. By induction assumption there is a function h_0 with the required time-complexity, and such that, if $\vec{\rho} \vdash\!\!\!\vdash_{\bar{P}} \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \vdash\!\!\!\vdash_{\bar{P}} ([\mathbf{t}/u]\psi)[\mathbf{w}, \vec{\mathbf{s}}]$. Let

$$h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \underset{\text{df}}{=} \langle \mathbf{t}[\mathbf{w}, \vec{\mathbf{s}}], h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \rangle.$$

∃-elim. $\Gamma \Rightarrow \varphi$ is derived from sequents $\Gamma \Rightarrow \exists u\psi$ and $\Gamma, [z/u]\psi \Rightarrow \varphi$. Here z is free in neither Γ nor φ, by the syntactic restriction on the ∃-elimination rule, so z is not among $u_1 \cdots u_k$; let z be u_{k+1}. The Subformula Lemma implies that $\exists u\psi$ is a positive subformula of Γ, and ψ is therefore either of the form $\mathbf{t} = \mathbf{cr}$ ($\mathbf{c} = \mathbf{0}$ or $\mathbf{1}$), or is a subformula of some α_i or β_j. In either case $\psi \in \Sigma_1^+$, and by the induction assumption there are functions $h_\exists$ and h_0 of the required time-complexity, such that:

(1) if $\vec{\rho} \Vdash_P \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_\exists(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_P (\exists u\psi)[\mathbf{w}, \vec{\mathbf{s}}]$, i.e.,

$$\mathrm{proj}_1\, h_\exists(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_P \psi[\mathbf{w}, \vec{\mathbf{s}}, \mathrm{proj}_0\, h_\exists(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})];$$

and

(2) for all $\mathbf{q} \in T$, if $\vec{\rho} \Vdash_P \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{q}]$, i.e. (since u_{k+1} is not free in $\vec{\beta}$), if $\vec{\rho} \Vdash_P \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}$, and $\tau \Vdash_P \psi[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{q}]$, then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \mathbf{q}, \vec{\rho}, \tau) \Vdash_P \varphi[\mathbf{w}, \vec{\mathbf{s}}, \mathbf{q}]$. Let

$$h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \underset{\mathrm{df}}{=} h_0(\mathbf{w}, \vec{\mathbf{s}}, \mathrm{proj}_0\, h_\exists(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}), \vec{\rho}, \mathrm{proj}_1\, h_\exists(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho})).$$

∀-intro. $\varphi \in \Sigma_1^+(Q)$ cannot be derived by ∀-introduction.

∀-elim. $\varphi \equiv [\mathbf{t}/u]\psi$ is derived from $\forall u\psi$. By the Subformula Lemma, $\forall u\psi$ must be a positive subformula of Γ, and since ψ is a Σ_1^+ formula, it must be a conjuct of $\tilde{\forall} P$. Then φ is an instance of an equation in P, and so putting $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) =_{\mathrm{df}} \varepsilon$ yields $h(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_P \varphi[\mathbf{w}, \vec{\mathbf{s}}]$ for all $\vec{\mathbf{s}}$.

abs. φ is derived from $\bot$. By induction assumption there is a function h_0 of the required time-complexity, such that if $\vec{\rho} \Vdash_P \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$, then $h_0(\mathbf{w}, \vec{\mathbf{s}}, \vec{\rho}) \Vdash_P \bot$. Since no object realizes $\bot$, this implies that there are no $\vec{\rho}$ such that $\vec{\rho} \Vdash_P \vec{\beta}[\mathbf{w}, \vec{\mathbf{s}}]$. We may therefore let $h = h_0$. ∎

RECEIVED December 5, 1991; FINAL MANUSCRIPT RECEIVED December 17, 1993

REFERENCES

[Bar69] BARWISE, J. (1969), Applications of strict Π_1^1 predicates to infinitary logic, *J. Symbolic Logic* **34**, 409–423.

[BC92] BELLANTONI, S., AND COOK, S. (1992), A new recursion-theoretic characterization of the poly-time functions, *Comput. Complexity* **2**, 97–110.

[Blo92] BLOCH, S. (1992), Functional characterizations of uniform log-depth and polylog-depth circuit families, *in* "Proc. 1992 IEEE Conference on Structure in Complexity," pp. 193–206, IEEE Computer Society Press, Washington, DC.

[Bus86] BUSS, S. (1986), "Bounded Arithmetic," Bibliopolis, Naples.

[Cob65] COBHAM, A. (1962), The intrinsic computational difficulty of functions, *in* "Proc. International Conference on Logic, Methodology, and Philosophy of Science" (Y. Bar-Hillel, Ed.), pp. 24–30, North-Holland, Amsterdam.

[Fef69] FEFERMAN, S. (1969), Systems of predicative analysis, *in* "The Philosophy of Mathematics" (J. Hintikka, Ed.), Oxford University Press.

[Fri75] FRIEDMAN, H. (1975), Some systems of second order arithmetic and their use, *in* "Proc. International Congress of Mathematicians I, Vancouver, 1974," pp. 235–242, Canadian Mathematical Congress.

[Gir71] GIRARD, J. Y. (1971), Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types, *in* "Proceedings of the Second Scandinavian Logic Symposium" (J. E. Fenstad, Ed.), pp. 63–92, North-Holland, Amsterdam.

[GS86] GUREVICH, Y. AND SHELAH, S. (1986), Fixed-pint extensions of first order logic, *Ann. Pure Appl. Logic* **32**, 265–280.

[GSS92] GIRARD, J. Y., SCEDROV, A., AND SCOTT, P. (1992), Bounded linear logic: A modular approach to polynomials time computability, *Theoret. Comput. Sci.* **97**, 1–66.

[Gur88] GUREVICH, Y. (1988), The logic and computer science column, *Bull. European Assoc. Theoret. Comput. Sci.* **35**, 71–82.

[Ign92] IGNJATOVIC, A. (1992), "Delineating Classes of Computational Complexity via Second Order Theories with Weak Set Existence Principles (I)," Tech. Rep. CMU-PHIL-22, Nov. 1991; *J. Symbolic Logic,* to appear.

[Imm86] IMMERMAN, N. (1986), Relational queries computable in polynomial time, *Information and Control* **68**, 86–104.

[Imm87] IMMERMAN, N. (1987), Languages which capture complexity classes, *SIAM J. Comput.* **16**, 760–778.

[Isl92] ISLES, D. (1992), What evidence is there that 2^65536 is a natural number?, *Notre Dame J. Formal Logic* **33**, 465–480.

[Kle52] KLEENE, S. C. (1952), "Introduction to Metamathematics," Van Nostrand, Princeton, NJ.

[Kle69] KLEENE, S. C. (1969), "Formalized Recursive Functions and Formalized Realizability," Memoirs of the AMS, Vol. 89, American Mathematical Society, Providence, RI.

[Kre60] KREISEL, G. (1960), La Prédicativité, *Bull. Soc. Math. France* **88**, 371–391.

[Kre60a] KREISEL, G. (1960), Status of the first epsilon-number in first order arithmetic, *J. Symbolic Logic* **25**, 390. [abstract]

[Kre65] KREISEL, G. (1965), Mathematical logic, *in* "Lectures on Modern Mathematics" (T. Saaty, Ed.), Vol. 3, pp. 95–195, Wiley, New York.

[Lei83] LEIVANT D. (1983), Reasoning about functional programs and complexity classes associated with type disciplines, *in* "Twenty-Fourth Annual Symposium on Foundations of Computer Science," pp. 460–469.

[Lei90]] LEIVANT, D. (1990), Contracting proofs to programs, *in* "Logic and Computer Science" (P. Odifreddi, Ed.), pp. 279–327, Academic Press, London.

[Lei90a] LEIVANT, D. (1990), Inductive definitions over finite structures, *Inform. and Comput.* **89**, 95–108.

[Lei90b] LEIVANT, D. (1990), Subrecursion and lambda representation over free algebras (Preliminary summary), *in* "Feasible Mathematics" (S. Buss and P. Scott, Eds.), pp. 281–291, Perspectives in Computer Science, Birkhauser, Boston/New York.

[Lei91] LEIVANT, D. (1991), A foundational delineation of computational feasibility, *in* "Proc. Sixth IEEE Conference on Logic in Computer Science," Amsterdam, IEEE Computer Society Press, Washington, DC.

[Lei92] LEIVANT, D. (1992), Semantic characterization of number theories, *in* "Logic from Computer Science" (Y. Moschovakis, Ed.), pp. 295–317, Springer-Verlag, Berlin.

[Lei93] LEIVANT, D. (1993), Stratified functional programs and computational complexity, *in* "Conference Record of the Twentieth Annual Symposium on Principles of Programming Languages," ACM, New York.

[Lei94] LEIVANT, D. (1994), Peano theories and the axiomatization of feasible mathematics, to appear.

[LM93] LEIVANT, D. AND MARION, J.-Y. (1993), Lambda characterizations of poly-time, *Fundamenta Informaticae* **19**, 167–184.

[Nel86] NELSON, E. (1986), "Predicative Arithmetic," Princeton Univ. Press, Princeton, NJ.

[Par77] PARSONS, C. (1977), On a number-theoretic choice schema and its relation to induction, *in* "Intuitionism and Proof Theory" (A. Kino *et al.*, Eds.), pp. 439–473, North-Holland, Amsterdam.

[Pea89] PEANO, G. (1889), "Arithmetices Principia, Novo Methodo Exposita," Turin, 1889. Engl. transl. *in* "From Frege to Gödel" (J. van Heijenoort, Ed.), pp. 83–97, Harvard Univ. Press, Cambridge, MA.

[Pra65] PRAWITZ, D. (1965), "Natural Deduction," Almqvist and Wiskel, Uppsala.

[Pra71] PRAWITZ, D. (1971), Ideas and results in proof theory, *in* "Proc. Second Scandinavian Logic Symposium" (J. E. Fenstad, Ed.), pp. 236–307, North-Holland, Amsterdam.

[Saz80] SAZONOV, V. (1980), Polynomial computability and recursivity in finite domains, *Electronische Informationsverarbeitung und Kybernetik* **7**, 319–323.

[Sch80] SCHÖNHAGE, A. (1980), Storage modification machines, *SIAM J. Comput.* **9**, 490–508.

[Sim86] SIMPSON, S. (1986), Subsystems of Z_2 and reverse mathematics (appendix), *in* "Proof Theory," Gaisi Takeuti, 2nd ed., pp. 434–448, North-Holland, Amsterdam.

[Tro73] TROELSTRA, A. S. (1973), "Metamathematical Investigation of Intuitionistic Arithmetic and Analysis," Lecture Notes in Math., Vol. 344, Springer-Verlag, Berlin.

[Var82] VARDI, M. (1982), Complexity and relational query languages, *in* "Fourteenth CM Symposium on Theory of Computing," pp. 137–146.